UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación en videojuegos de técnicas de planificación con simuladores

TRABAJO FIN DE MASTER

Master Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Cristóbal Cervantes Rodríguez

Tutor: Eva Onaindia de la Rivaherrera

Curso 2018-2019

Resum

????

Paraules clau: ????, ?????????, ????, ??????????????

Resumen

????

Palabras clave: ?????, ???, ??????????????

Abstract

????

Key words: ?????, ????? ?????, ?????????????

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Objectives/Motivation

????? ????????????? ????????????? ????????????? ????????????? ?????????????

CAPÍTULO 2

Related Work

????? ????????????? ????????????? ????????????? ????????????? ?????????????

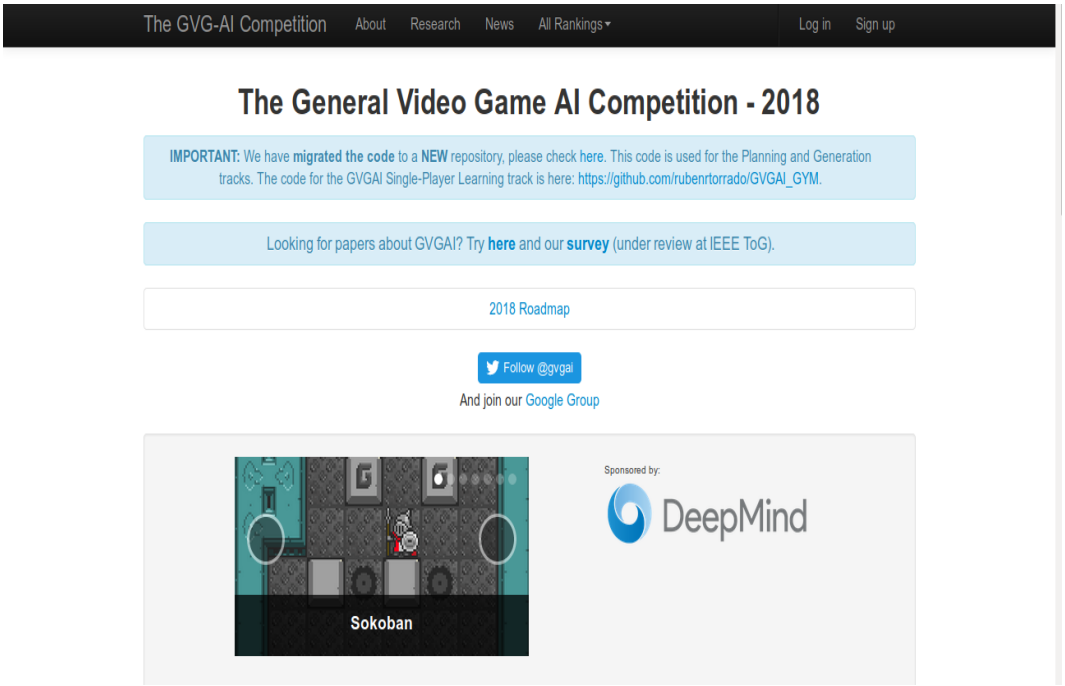
CAPÍTULO 3
Background

In this chapter we are going to detail all those elements to understand this proyect. First we introduce "The General Video Game AI Competition(GVG-AI) [3], where all the rules that govern our agent are defined. Next we detail the most importants concepts about planning implementations, focusing in the algorithm chosen for our agent.

3.1 GVG-AI

General Video Game Playing (GVGP) is a sub-domain of Game Artificial Intelligence which aims to create generic enough agents, capable of playing any given game, without pre-computed game-specific heuristics, in possibly unknown environments. It is meant to put complex algorithms to the test, challenging their adaptibility to new situations. There are differents approaches such as deep reinforcement learning combined with Monte Carlo Tree Search which can be engi- neered to do brilliantly in some specific games. [1]

The GVG-AI Competition explores the problem of creating controllers for general video game playing. This controllers must be able to play different games without previous knowledge about them.



The framework of this competition show us information about the games in an object-oriented manner, and employs a Video Game Description Language (VGDL) 3.1.3 to define games in a more general way.

In this enviroment, the agent receives calls at every game step and must return a discrete action to apply in no more than 40ms. In this period we need explore the different paths or combinations of actions to obtain the best rewards and win the games. That limit, combined with the complexity of the problem difficult us to explore the differents combinations and for this reason we need to use an strategy that could obtain a good option and avoid "the Game Over".

It counts on one second for initialization at the beginning of the game too. Additionally, the engine does not provide the VGDL description of the game, nor the rules or ways to achieve victory, leaving this to the agent to discover. Our player can obtain information about the state e.g. positions, resources, score. Also agent can make simulations about next iterations. With this information, it should discover the game mechanics and do its best in each game.

3.1.1. Filosofy

????? ????????????? ????????????? ????????????? ????????????? ?????????????

3.1.2. Simulator

By means of the 'State Observation', the agent can comunicate with the game world. Throw this object we can interact to extract information about the actual state. Besides we can make simulations about futures movements with which we could make decisions about the best combinations to obtain our best score.

Said object contains methods with which to determine, the position and characteristics of all the "sprites.ºf the game. Being able to differentiate several large groups.

- Non-player-characters (NPC): It's composed by that characters that interfere in the game excepting the principal character that is controlled by us. The could be enemies that try to hurt us or sprites that we should defend.
- Inamobibles: Formed by all those sprites that had a fixed position in our grid. Mainly contain sprites that form the ground, the walls, obstacles and other elements that remain static throughout the course of the game.
- Movable: Included in this group are those elements that can be moved and are not included in the NPC set. They can vary greatly depending on the game, but include bullets, boxes that can be moved by our avatar, and many others.
- Resources: That list contain the resources that can be used by our avatar.
- Portals: In this list are included the positions of that elements in the game able to transport us between distant positions of the game, as well as to create or to make disappear other sprites.

Sprites have differents behaviors depending of the game and could vary a lot depending of game. It made almost impossible to extrapolate rules for play in all games. Being necessary to use heuristics that allow discover the mechanics of the game and find the best actions in each step.

Through the state observation, agent know the possible actions that can choose in each game. That set could vary throughout the games too. For example in the game "space invaders" agent only can displace in two directions and can also use the action_use action to shot and kill its enemies.

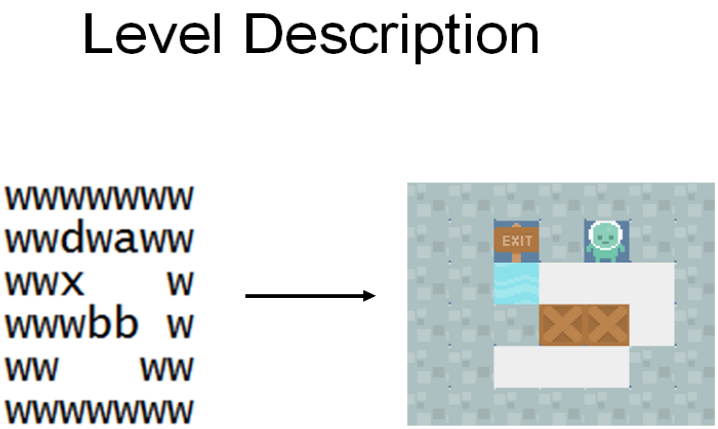
The following are the set of possible actions in each game:

- ACTION_NIL: It's a reserved action used by the system when the agent dont choose an action in the time provided previously to disqualify.
- ACTION_UP: Mainly permit to the agent to move or to orientate upwards.
- ACTION_LEFT: Mainly permit to the agent to move or to orientate to the left.
- ACTION_DOWN: Mainly permit to the agent to move or to orientate downwards.
- ACTION_RIGH: Mainly permit to the agent to move or to orientate to the right.
- ACTION_USE: It is the most versatile action. Its result depends on each game although in many of the games it is not operative

Our principal task is to determine, the best action of that set to be applied in each step. For help us, the framework let us to make simulations in each step. And query to the state observation about the new situation for determine which action is more profitable. We can make those simulations iteratively for explore large paths applying searching algorithms with differents heuristics in a given time. Besides, some games have a non-deterministic behaviour. We should have took it into account for make our simulations to obtain a safe route.

3.1.3. VGDL

Thanks to the simulator, we can see and play the available games, but what allows our agent to be able to play in all the games is a common language to be created, represented and interpreted for our simulator. The Video Game Definition Language (VGDL) is the language chossed by this task. It's a language design by Tom Schaul, originally implemented by him in Python using py-game.



2D matrix of symbols

It's a simple, high-level description language for 2D video games that permit a library to parse and play those games. The streamlined design of the language is based on defining locations and dynamics for simple building blocks, and the interaction effects when such objects collide, all of which are provided in a rich ontology. [4]

It arises as a need for a way of representing games with certain characteristics. The language should be clear, human-readable, and unambiguous. Its vocabulary should be highly expressive from the beginning, yet still extensible to novel types of games. Finally, its representation structure should be easy to parse and facilitate automatically generated games, in such a way that default settings and sanity checks enable most random game description to be actually playable.

From all this emerged this language capable of creating a game in a few lines. An example is shown in the next picture where is representing the game zelda.

```
BasicGame
  LevelMapping
    G > goal
    + > key
    A > nokey
    1 > monster
  SpriteSet
    goal   > Immovable color=GREEN
    key    > Immovable color=ORANGE
    sword > Flicker limit=5 singleton=True
    movable      >
      avatar     > ShootAvatar stype=sword
        nokey    >
        withkey > color=ORANGE
      monster    > RandomNPC cooldown=4
  InteractionSet
    movable wall   > stepBack
    nokey goal     > stepBack
    goal withkey   > killSprite
    monster sword > killSprite
    avatar monster> killSprite
    key  avatar    > killSprite
    nokey key   > transformTo stype=withkey
  TerminationSet
    SpriteCounter stype=goal   win=True
    SpriteCounter stype=avatar win=False
```

The level description is simply a text string/file with four differnced blocks:

- The LevelMapping permit read the level description to generate the initial state, transforming each character in the correspondings sprites.
- The SpriteSet section defines all the ontology of sprites. Using nested indentations we define the hierarchy where nested elements share the objects properties of their parents. We can augmented the definition of a sprite class adding keywords options in its definition.
- The InteractionSet governs the transitions between states, especially in relation to collisions between sprites. Here we define wich sprites are destroyed or created or when a sprite can't move to a postion grid for be occupied by another sprite.
- The TerminationSet defines different ways by wich the game can end. Each termination instance defines if game end with our victory or in a game over.

3.2 IW

In this section we will explain the main concepts of the most used algorithms in videogame planning.

3.2.1. Planing concepts

When create an heuristic or an agent to play a game, the objective is usually to win or to maximize the game score. When we are developing an agent for play in a game, we can

learn about the dynamics of the game and transmit all this knowledge to our agent. For example if the game require that we collect all objects of some type, we could add some restrictions for look for the best way to obtain it. Or if in one puzzle game, we move a box beside a wall, then we will not be able to separate it. For this reason, if we were realizing an agent for that type of games, we would include in his heuristic, some restriction to avoid that type of movements unless we found the solution.

Instead, in the general video game context our main is make heuristics capable off play in differents types of games. In this context we should be careful adding beneficial rules for a type of game because they can be harmful in other contexts. We need heuristics more generics that provide good global results.

In this area the approaches more used are in agreement with te agents observed in the last years in the GVGAI competition. Their agents used from Evolutionary Algorithms (EA) to MonteCarlo Tree Search (MCTS) techniques, including multiplevariants and other more straightforward approaches such asBreadth First Search (BFS) or A*. In some cases, several ofthese approaches are glued together, using a high level switchthat determines the algorithm to use, in the form of a meta-heuristic[2].

This approaches can be divided in two big groups

On one side, there are approaches that use a mapping between the input game and the action executed in each state. The phylosophy in that kind of implementations is that the agent can learn in each disposition of the grid observations, what is the best action and play in consequently. The principal implementation of this type is make a value-function using neural networks to encode a mapping from observations to actions.

This approach is very useful if the state-space is large. Usually the training of the neural model it's very expensive so it can not be done in run time. It's necessary create the model previously trained in large number of games and executions. The big advantage it's that in execution time it's faster and not need to explore a big number of nodes to acquire a knowledge about the game.

On the contrary a change in the input or a new game with big differences could make the model completely ineffective. And will be made necessary retrain all the model.

On the other hand, there are algorithms based on search approach in which each iteration generate all search tree to obtain the better action that we have been able to find.

In this group we can situate the IW algorithm that we explain more deeply in the next section.

Other tipical used algorithm in this general video games is Monte Carlo Tree Search (MTCS). It's a reinforcement learning approach that has become very popular in the last decade due to its performance in the game of Go. It' s a tree based search algorithm that builds an asymmetric tree in memory using a forward model of the game. The algorithm adds one node at each iteration, estimating its game-theoretic value by using self-play from the state of the node to the end of the game or a given depth.

3.2.2. Algorithm IW

????? ????????????? ????????????? ????????????? ????????????? ?????????????

CAPÍTULO 4

New novelty-based technique for videogames

????? ????????????? ????????????? ????????????? ????????????? ?????????????

4.1 preproceso

????? ????????????? ????????????? ????????????? ????????????? ?????????????

4.2 Abstracción de estados

????? ????????????? ????????????? ????????????? ????????????? ?????????????

4.3 IW

????? ????????????? ????????????? ????????????? ????????????? ?????????????

4.3.1. adaptacion

????? ????????????? ????????????? ????????????? ????????????? ?????????????

4.3.2. Implementation details

????? ????????????? ????????????? ????????????? ????????????? ?????????????

CAPÍTULO 5
Results

Bibliografía

[1] Jennifer S. Light. When computers were women. *Technology and Culture*, 40:3:455–483, juliol, 1999.

[2] Georges Ifrah. *Historia universal de las cifras*. Espasa Calpe, S.A., Madrid, sisena edició, 2008.

[3] Comunicat de premsa del Departament de la Guerra, emés el 16 de febrer de 1946. Consultat a http://americanhistory.si.edu/comphist/pr1.pdf.

[4] The General Video Game AI Competition

http://www.gvgai.net.

Bibliografía

[1] Raluca D Gaina, Diego Pérez-Liébana, and Simon M Lucas. General video game for 2 players: framework and competition. In *Computer Science and Electronic Engineering (CEEC), 2016 8th*, pages 186–191. IEEE, 2016.

[2] Cristina Guerrero-Romero, Annie Louis, and Diego Perez-Liebana. Beyond playing to win: Diversifying heuristics for gvgai. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, pages 118–125. IEEE, 2017.

[3] MultiMedia LLC. The gvg-ai competition.

[4] Tom Schaul. A video game description language for model-based or interactive learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, pages 1–8. IEEE, 2013.

APÉNDICE A
Configuració del sistema

????? ????????????? ????????????? ????????????? ????????????? ?????????????

A.1 Fase d'inicialització

????? ????????????? ????????????? ????????????? ????????????? ?????????????

A.2 Identificació de dispositius

????? ????????????? ????????????? ????????????? ????????????? ?????????????

APÉNDICE B
??? ????????????? ????

????? ????????????? ????????????? ????????????? ????????????? ?????????????